

CSE 414 Database Project Report
Yakup Talha Yolcu
1801042609

TRANSPORTATION AND ACCOMMODATION
BOOKING SYSTEM

CONTENTS

CONTENTS	2
PROBLEM DEFINITION	3
TRANSPORTATION AND ACCOMMODATION BOOKING SYSTEM DESCRIPTION	3
IMPLEMENTATION DETAILS	3
USER REQUIREMENTS	4
Person	4
Guide	4
Company	4
ER DIAGRAM	5
TABLES	7
SELECT EXAMPLES	9
INSERT EXAMPLE - 1	12
INSERT EXAMPLE - 2	13
INSERT EXAMPLE - 3	13
UPDATE EXAMPLE - 1	14
UPDATE EXAMPLE - 2	15
UPDATE EXAMPLE - 3	16
DELETE EXAMPLE - 1	17
DELETE EXAMPLE - 2	18
DELETE EXAMPLE - 3	19
FUNCTIONAL DEPENDENCIES	20
CHECKING NORMALIZATION	21
VIEWS	23
TRIGGERS	24
Update person budget when accommodation is bought or canceled	24
Update person budget when car rental is bought or canceled	25
Update person budget when a transport is bought or canceled	25
Can't rent a car more than one week and check budget is enough	28
Cant accommodate on a hotel more than one month and check budget is enough	29
Check budget for transportation	30
JOINS	31
RIGHT JOIN	31
LEFT JOIN	32
FULL JOIN	33
INNER JOIN	33
ATOMIC TRANSACTIONS	35
While person adds a budget	35
While person removes a budget	36
While company transfers a guide	37
While company transfers a car_rental	38

PROBLEM DEFINITION

The main purpose of this project is manage the database of a travel agency efficiently. Agencies can manage tours, flights, bus trips, accommodations for persons, car rental for persons. With my experience , I did not see a system like that. This system combines all travel and accommodation occasions together. System will reduce the sightseers burden.

Companies will get benefit also, they do not need to track for each person. With tours, companies can combine flight or bus trips with an accommodation.

TRANSPORTATION AND ACCOMMODATION BOOKING SYSTEM DESCRIPTION

In this project, the database implementation and interface of a transportation and accommodation booking system is made. First of all, system can be used by 3 users.

- person
- guide
- company

Persons can be added by company with person_id, name, birthdate and a budget. Guides can be also added by company. Guides are also persons. There is an inheritance. Guides can do whatever person do. In addition to that, guides can view assigned transportations them. Company offers these functionalities :

- plan bus trip
- plan flight trip
- plan accommodation on a hotel
- renting a car
- joining a tour (bus or flight + accommodation)

IMPLEMENTATION DETAILS

I used MYSQL for database and Java for development. I used Java Swing Library for pretty UI.

USER REQUIREMENTS

Person

Person can do :

- View / Buy / Cancel Car rentals
- View / Buy / Cancel Flight trip
- View / Buy / Cancel Bus trip
- View / Buy / Cancel Accommodation
- View tour
- Cancel tour joining

Guide

Guide can do :

- Do whatever person can do
- View guided transports

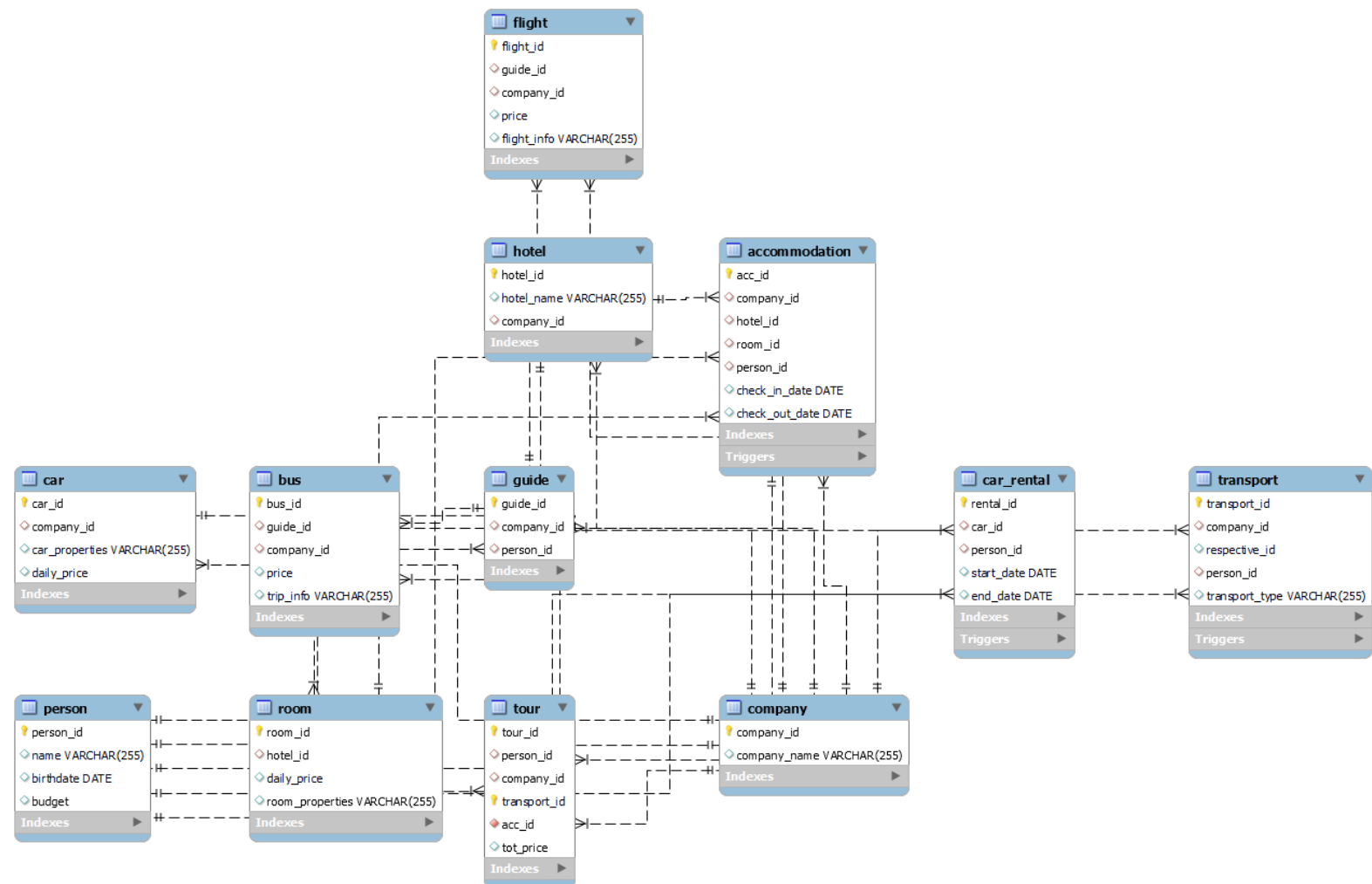
Company

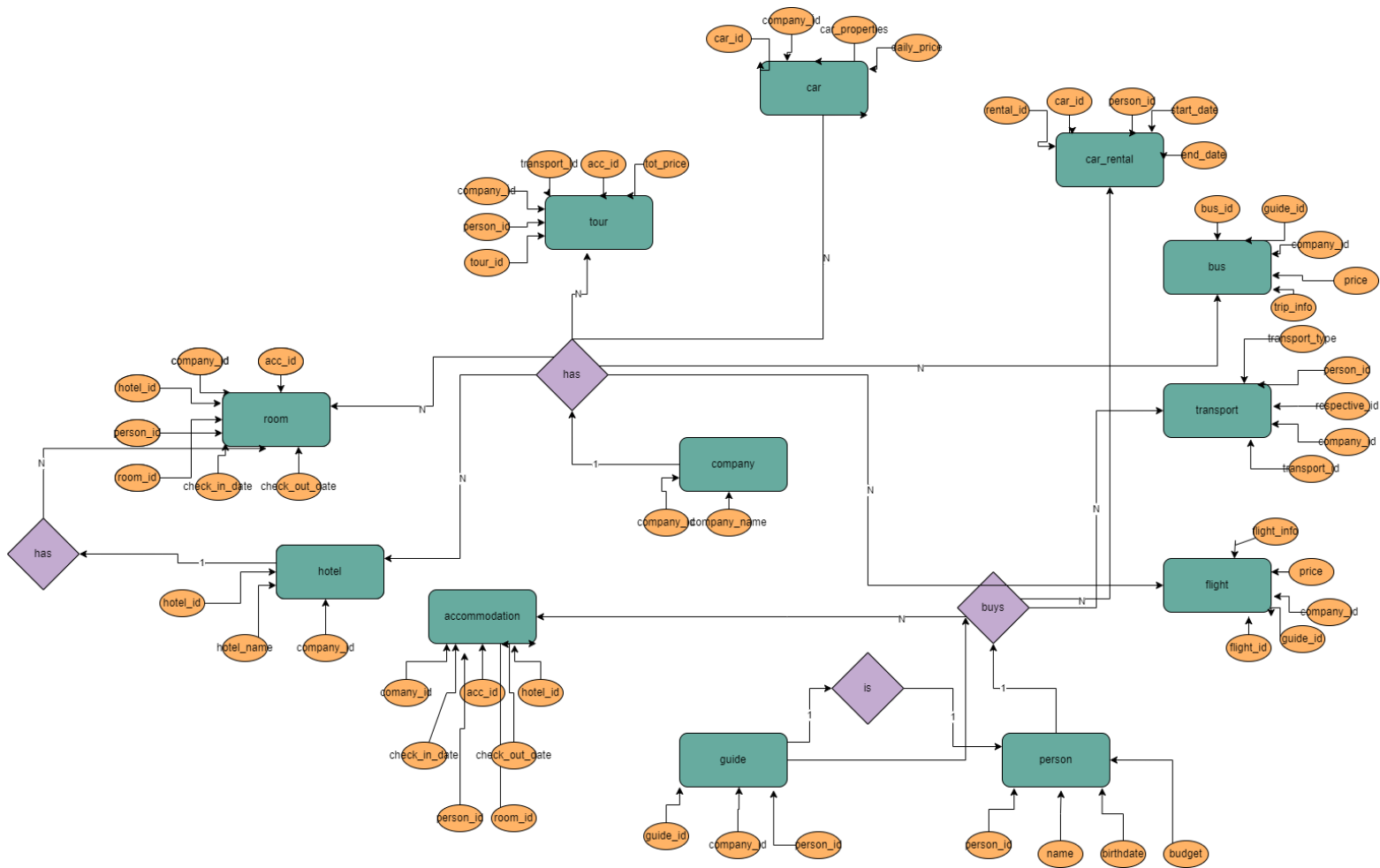
Company can do :

- Insert / Delete / Update car and car rental
- Insert / Delete / Update person
- Insert / Delete / Update accommodation, hotel and room
- Insert / Delete / Update flight, bus trip and its transportations
- Insert / Delete / Update tours
- Insert / Delete / Update guides

And to manage companies, I've added Insert / Delete / Update company.

ER DIAGRAM





TABLES

```
1 CREATE TABLE `accommodation` (  
2   `acc_id` int unsigned NOT NULL AUTO_INCREMENT,  
3   `company_id` int DEFAULT NULL,  
4   `hotel_id` int DEFAULT NULL,  
5   `room_id` int DEFAULT NULL,  
6   `person_id` int DEFAULT NULL,  
7   `check_in_date` date DEFAULT NULL,  
8   `check_out_date` date DEFAULT NULL,  
9   PRIMARY KEY (`acc_id`),  
10  UNIQUE KEY `acc_id_UNIQUE` (`acc_id`),  
11  KEY `company_id` (`company_id`),  
12  KEY `hotel_id` (`hotel_id`),  
13  KEY `room_id` (`room_id`),  
14  KEY `person_id` (`person_id`),  
15  CONSTRAINT `accommodation_ibfk_1` FOREIGN KEY (`company_id`) REFERENCES `company` (`company_id`),  
16  CONSTRAINT `accommodation_ibfk_2` FOREIGN KEY (`hotel_id`) REFERENCES `hotel` (`hotel_id`),  
17  CONSTRAINT `accommodation_ibfk_3` FOREIGN KEY (`room_id`) REFERENCES `room` (`room_id`),  
18  CONSTRAINT `accommodation_ibfk_4` FOREIGN KEY (`person_id`) REFERENCES `person` (`person_id`)  
19 ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci  
20  
21 CREATE TABLE `bus` (  
22   `bus_id` int NOT NULL,  
23   `guide_id` int DEFAULT NULL,  
24   `company_id` int DEFAULT NULL,  
25   `price` int DEFAULT NULL,  
26   `trip_info` varchar(255) DEFAULT NULL,  
27   PRIMARY KEY (`bus_id`),  
28   KEY `guide_id` (`guide_id`),  
29   KEY `company_id` (`company_id`),  
30   CONSTRAINT `bus_ibfk_1` FOREIGN KEY (`guide_id`) REFERENCES `guide` (`guide_id`),  
31   CONSTRAINT `bus_ibfk_2` FOREIGN KEY (`company_id`) REFERENCES `company` (`company_id`)  
32 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci  
33
```

```
34 CREATE TABLE `car` (  
35   `car_id` int NOT NULL,  
36   `company_id` int DEFAULT NULL,  
37   `car_properties` varchar(255) DEFAULT NULL,  
38   `daily_price` int DEFAULT NULL,  
39   PRIMARY KEY (`car_id`),  
40   KEY `company_id` (`company_id`),  
41   CONSTRAINT `car_ibfk_1` FOREIGN KEY (`company_id`) REFERENCES `company` (`company_id`)  
42 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci  
43  
44 CREATE TABLE `car_rental` (  
45   `rental_id` int NOT NULL AUTO_INCREMENT,  
46   `car_id` int DEFAULT NULL,  
47   `person_id` int DEFAULT NULL,  
48   `start_date` date DEFAULT NULL,  
49   `end_date` date DEFAULT NULL,  
50   PRIMARY KEY (`rental_id`),  
51   KEY `car_id` (`car_id`),  
52   KEY `person_id` (`person_id`),  
53   CONSTRAINT `car_rental_ibfk_1` FOREIGN KEY (`car_id`) REFERENCES `car` (`car_id`),  
54   CONSTRAINT `car_rental_ibfk_2` FOREIGN KEY (`person_id`) REFERENCES `person` (`person_id`)  
55 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci  
56  
57 CREATE TABLE `company` (  
58   `company_id` int NOT NULL,  
59   `company_name` varchar(255) DEFAULT NULL,  
60   PRIMARY KEY (`company_id`)  
61 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci  
62
```

```

63 CREATE TABLE `flight` (
64     `flight_id` int NOT NULL,
65     `guide_id` int DEFAULT NULL,
66     `company_id` int DEFAULT NULL,
67     `price` int DEFAULT NULL,
68     `flight_info` varchar(255) DEFAULT NULL,
69     PRIMARY KEY (`flight_id`),
70     KEY `guide_id` (`guide_id`),
71     KEY `company_id` (`company_id`),
72     CONSTRAINT `flight_ibfk_1` FOREIGN KEY (`guide_id`) REFERENCES `guide` (`guide_id`),
73     CONSTRAINT `flight_ibfk_2` FOREIGN KEY (`company_id`) REFERENCES `company` (`company_id`)
74 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
75
76 CREATE TABLE `guide` (
77     `guide_id` int NOT NULL,
78     `company_id` int DEFAULT NULL,
79     `person_id` int DEFAULT NULL,
80     PRIMARY KEY (`guide_id`),
81     KEY `company_id` (`company_id`),
82     KEY `person_id` (`person_id`),
83     CONSTRAINT `guide_ibfk_1` FOREIGN KEY (`company_id`) REFERENCES `company` (`company_id`),
84     CONSTRAINT `guide_ibfk_2` FOREIGN KEY (`person_id`) REFERENCES `person` (`person_id`)
85 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
86
87 CREATE TABLE `hotel` (
88     `hotel_id` int NOT NULL,
89     `hotel_name` varchar(255) DEFAULT NULL,
90     `company_id` int DEFAULT NULL,
91     PRIMARY KEY (`hotel_id`),
92     KEY `company_id` (`company_id`),
93     CONSTRAINT `hotel_ibfk_1` FOREIGN KEY (`company_id`) REFERENCES `company` (`company_id`)
94 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
95

```

```

CREATE TABLE `person` (
    `person_id` int NOT NULL,
    `name` varchar(255) DEFAULT NULL,
    `birthdate` date DEFAULT NULL,
    `budget` int DEFAULT NULL,
    PRIMARY KEY (`person_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `room` (
    `room_id` int NOT NULL,
    `hotel_id` int DEFAULT NULL,
    `daily_price` int DEFAULT NULL,
    `room_properties` varchar(255) DEFAULT NULL,
    PRIMARY KEY (`room_id`),
    KEY `hotel_id` (`hotel_id`),
    CONSTRAINT `room_ibfk_1` FOREIGN KEY (`hotel_id`) REFERENCES `hotel` (`hotel_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `tour` (
    `tour_id` int NOT NULL,
    `person_id` int DEFAULT NULL,
    `company_id` int DEFAULT NULL,
    `transport_id` int unsigned NOT NULL,
    `acc_id` int unsigned NOT NULL,
    `tot_price` int DEFAULT NULL,
    PRIMARY KEY (`tour_id`,`transport_id`),
    KEY `person_id` (`person_id`),
    KEY `company_id` (`company_id`),
    KEY `ibfk4` (`transport_id`),
    KEY `ibfk5` (`acc_id`),
    CONSTRAINT `ibfk5` FOREIGN KEY (`acc_id`) REFERENCES `accommodation` (`acc_id`),
    CONSTRAINT `tour_ibfk_1` FOREIGN KEY (`person_id`) REFERENCES `person` (`person_id`),
    CONSTRAINT `tour_ibfk_2` FOREIGN KEY (`company_id`) REFERENCES `company` (`company_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```


SELECT EXAMPLES

Accommodation Entries

acc_id	company...	hotel_id	room_id	person_id	check_in...	check_ou...
4	1	1	1	1	2023-06-...	2023-06-...

```
// Construct the query to retrieve accommodations for the specified company
String query = "SELECT * FROM accommodation WHERE company_id = " + companyID;
ResultSet resultSet = st.executeQuery(query);
```

Insert

Update

Delete

OK

Car Entries

car_id	company_id	car_properties	daily_price
1	1	Blue	200
2	1	Red	200
3	1	Green	200

```
String query = "SELECT * FROM car WHERE company_id = "+company_id;
ResultSet resultSet = st.executeQuery(query);
```

Insert

Update

Delete

OK

Car Rental Entries

rental_id	car_id	person_id	start_date	end_date
3	3	1	2023-06-08	2023-06-08

```
String query = "SELECT DISTINCT car_rental.rental_id, car_rental.car_id, car_rental.person_id, car_rental.start_date, car_rental.end_date " +
    "FROM car_rental " +
    "RIGHT JOIN car ON car.car_id = car_rental.car_id " +
    "WHERE car.company_id = " + company_id;
```

Insert

Update

Delete

OK

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8200
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

```
// Execute the SQL query to retrieve all person entries
String query = "SELECT * FROM person";
ResultSet resultSet = st.executeQuery(query);
```

Insert

Update

Delete

OK

Hotel Entries

hotel_id	hotel_name	company_id
1	Hotel1	1
2	Hotel2	1

```
// Construct the query to retrieve hotels for the specified company
String query = "SELECT * FROM hotel WHERE company_id = " + companyID;
ResultSet resultSet = st.executeQuery(query);
```

Insert

Update

Delete

OK

Room Entries

room_id	hotel_id	daily_price	room_properties
1	1	200	big
2	1	250	little

```
String query = "SELECT room.room_id, room.hotel_id, room.daily_price, room.room_properties "
    "FROM room " +
    "LEFT JOIN hotel ON room.hotel_id = hotel.hotel_id " +
    "WHERE room.hotel_id = "+hotelID;
```

Insert

Update

Delete

OK

Flight Entries

flight_id	guide_id	company_id	price	flight_info
1	1	1	200	ankara to ista...

```
Flight selectFlight = new Flight();
String selectFlightCompany=selectFlight.getSelectQueryCompany(Integer.parseInt(companyID));
ResultSet resultSet = st.executeQuery(selectFlightCompany);
```

1 usage Yakup Talha Yolcu

```
public String getSelectQueryCompany(int id) {
    return "SELECT * FROM flight WHERE company_id = "+id;
}
```

Insert

Update

Delete

OK

Bus Entries

bus_id	guide_id	company_id	price	trip_info
1	1	1	200	istanbul to an...

```
// Construct the query to retrieve buses for the specified company
String query = "SELECT * FROM bus WHERE company_id = " + companyID;
ResultSet resultSet = st.executeQuery(query);
```

Insert

Update

Delete

OK

Transport Entries

transport_id	company_id	respective_id	person_id	transport_type
4	1	1	1	flight

```
// Query the transport table to retrieve the transport entries for the specified company ID
String query = "SELECT * FROM transport WHERE company_id = " + companyId + ";";
ResultSet resultSet = st.executeQuery(query);
```

InsertUpdateDelete

OK

Company Entries

Company ID	Company Name
1	COMP1

```
// Execute the query to fetch company entries
String query = "SELECT * FROM company";
ResultSet resultSet = st.executeQuery(query);
```

InsertUpdateDelete

OK

View Tours

Tour ID	Person ID	Company ID	Accommod...	Transport ID	Total Price
1	1	1	4	4	500

```
// Execute the query to fetch tour entries for the specified company ID
String query = "SELECT * FROM tour WHERE company_id = " + companyId;
ResultSet resultSet = st.executeQuery(query);
```

InsertUpdateDelete

OK

View Guides

guide_id	company_id	person_id
1	1	2

```
// Execute the query to retrieve the guides for the specified company ID
String query = "SELECT * FROM guide WHERE company_id = " + companyId + ";";
ResultSet resultSet = st.executeQuery(query);
```

InsertUpdateDelete

OK

INSERT EXAMPLE - 1

Company inserts a person

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8400
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

Insert Update Delete

OK

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8400
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

Insert Person

? Person ID: 5

First Name: Julia

Birthdate: 2000-07-20

Budget: 2500

OK Cancel

Insert Update Delete

OK

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8400
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000
5	Julia	2000-07-20	2500

```
Person person = new Person(Integer.parseInt(personId), firstName, date, Integer.parseInt(budget));  
String query = person.getInsertQuery(person);
```

```
@Override  
public String getInsertQuery(Person object) {  
    return "INSERT INTO person (person_id, name, birthdate, budget) VALUES (" +  
        "\"" + object.getPerson_id() + "\", " +  
        "\"" + object.getName() + "\", " +  
        "\"" + object.getBirthdate() + "\", " +  
        "\"" + object.getBudget() +  
        "\");";  
}
```

Insert Update Delete

OK

INSERT EXAMPLE - 2

Company inserts bus
Enter company id

View Bus

Enter company_id:

1

OK Cancel

Enter bus table info

Before insert

Bus Entries

bus_id	guide_id	company_id	price	trip_info
1	1	1	200	istanbul to an...

While inserting :

Insert Bus

Bus ID: 2

Guide ID: 1

Company ID: 1

Price: 250

Trip Info: ankara to istanbul

OK Cancel

Insert Bus

Bus inserted successfully.

OK

```
Bus bus = new Bus(  
    Integer.parseInt(busId), Integer.parseInt(guideId),  
    Integer.parseInt(companyId),  
    Integer.parseInt(price), tripInfo  
);  
  
String insertQuery = bus.getInsertQuery(bus);
```

```
@Override  
public String getInsertQuery(Bus object) {  
    return "INSERT INTO bus (bus_id, guide_id, company_id, price, trip_info) VALUES (" +  
        "\"" + object.getBus_id() + "\", " +  
        "\"" + object.getGuide_id() + "\", " +  
        "\"" + object.getCompany_id() + "\", " +  
        "\"" + object.getPrice() + "\", " +  
        "\"" + object.getTrip_info() +  
        "\")";  
}
```

After insert :

Bus Entries

bus_id	guide_id	company_id	price	trip_info
1	1	1	200	istanbul to an...
2	1	1	250	ankara to ista...

INSERT EXAMPLE - 3

Company inserts hotel
Enter company_id

Before insert

View Hotel

Enter company_id:

1

OKCancel

Hotel Entries

hotel_id	hotel_name	company_id
1	Hotel1	1
2	Hotel2	1

While inserting :

Insert Hotel

Hotel ID: 3

Name: Hotel3

Company ID: 1

OKCancel

```
Hotel hotel = new Hotel(Integer.parseInt(hotelId),name,Integer.parseInt(companyId));  
String query = hotel.getInsertQuery(hotel);
```

Message

Insertion successful

OK

```
@Override  
public String getInsertQuery(Hotel object) {  
    return "INSERT INTO hotel (hotel_id, hotel_name, company_id) VALUES (" +  
        object.getHotel_id() + "," +  
        object.getHotel_name() + "," +  
        object.getCompany_id() +  
        ")";  
}
```

After insert :

Hotel Entries

hotel_id	hotel_name	company_id
1	Hotel1	1
2	Hotel2	1
3	Hotel3	1

UPDATE EXAMPLE - 1

Company updates a car

Car Entries

car_id	company_id	car_properties	daily_price
1	1	Blue	200
2	1	Red	200
3	1	Black	250

Insert

Update

Delete

OK

Car Entries

car_id	company_id	car_properties	daily_price
1	1	Blue	200
2	1	Red	200
3	1	Black	250

Enter car info

?

car_id

3

car_properties

White

daily_price

300

OK

Cancel

Insert

Update

Delete

OK

Car Entries

car_id	company_id	car_properties	daily_price
1	1	Blue	200
2	1	Red	200
3	1	White	300

Insert

Update

Delete

OK

```
Car car=new Car(Integer.parseInt(car_id.getText()),Integer.parseInt(company_id),car_properties.getText(),Integer.parseInt(daily_price.getText()));
String query = car.getUpdateQuery(car,Integer.parseInt(car_id.getText()));
```

```
@Override
public String getUpdateQuery(Car object, int id) {
    return "UPDATE car SET " +
        "car_id = " + object.getCar_id() + ", " +
        "company_id = " + object.getCompany_id() + ", " +
        "car_properties = '" + object.getCar_properties() + "', " +
        "daily_price = " + object.getDaily_price() +
        " WHERE car_id = " + id;
}
```

Insert

Update

Delete

OK

UPDATE EXAMPLE - 2

Updating a person
Before update

Person Entries			
person_id	name	birthdate	budget
1	Talha	2000-07-12	6600
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000
5	Julia	2000-07-20	2500

Update Person

?

Person ID:

5

First Name:

Julia

Birthdate:

2000-07-20

Budget:

2600

OK

Cancel

```
Person person = new Person(Integer.parseInt(personId), updatedFirstName, Date.valueOf(updatedAge), Integer.parseInt(updatedBudget));
String query = person.getUpdateQuery(person, Integer.parseInt(personId));
System.out.println(query);
```

```
@Override
public String getUpdateQuery(Person object, int id) {
    return "UPDATE person SET " +
        "person_id = " + object.getPerson_id() + ", " +
        "name = '" + object.getName() + "', " +
        "birthdate = '" + object.getBirthdate() + "', " +
        "budget = " + object.getBudget() +
        " WHERE person_id = " + id;
}
```

Message

i

Update successful

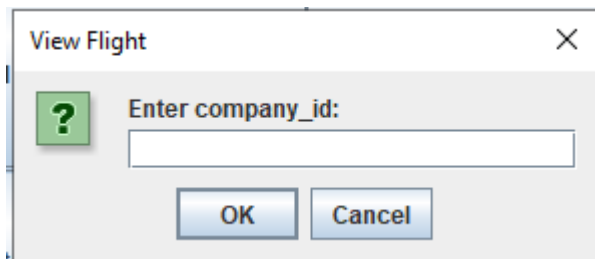
OK

After update

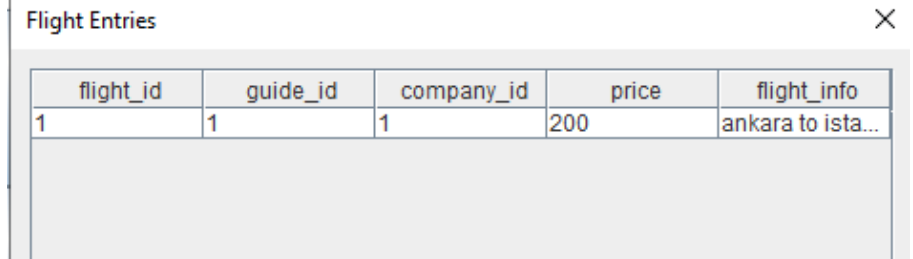
Person Entries			
person_id	name	birthdate	budget
1	Talha	2000-07-12	6600
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000
5	Julia	2000-07-20	2600

UPDATE EXAMPLE - 3

Company updates flight
Enter company_id



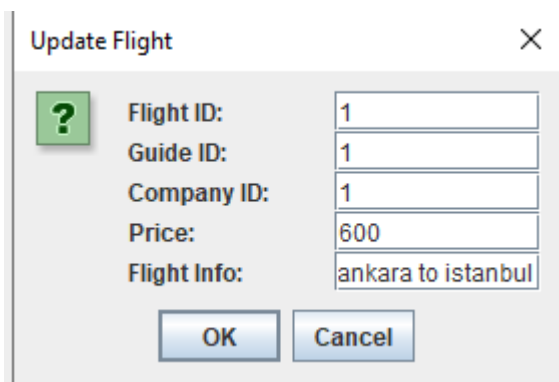
A dialog box titled "View Flight" with a close button (X) in the top right corner. It contains a green question mark icon, a label "Enter company_id:", a text input field, and two buttons labeled "OK" and "Cancel" at the bottom.



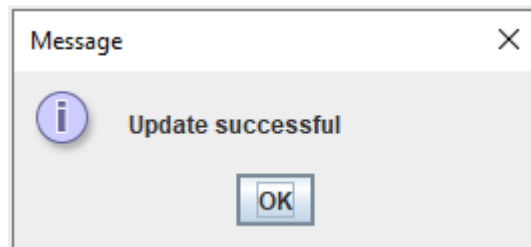
A table titled "Flight Entries" with a close button (X) in the top right corner. It contains one row of data.

flight_id	guide_id	company_id	price	flight_info
1	1	1	200	ankara to ista...

While updating :



A dialog box titled "Update Flight" with a close button (X) in the top right corner. It contains a green question mark icon and five labeled text input fields: "Flight ID:" (1), "Guide ID:" (1), "Company ID:" (1), "Price:" (600), and "Flight Info:" (ankara to istanbul). At the bottom are "OK" and "Cancel" buttons.



A message dialog box titled "Message" with a close button (X) in the top right corner. It contains an information icon (i) and the text "Update successful". At the bottom is an "OK" button.

After update :



A table titled "Flight Entries" with a close button (X) in the top right corner. It contains one row of data, where the price has been updated from 200 to 600.

flight_id	guide_id	company_id	price	flight_info
1	1	1	600	ankara to ista...

```
Flight flight = new Flight(  
    Integer.parseInt(updatedFlightId), Integer.parseInt(updatedGuideId),  
    Integer.parseInt(updatedCompanyId), Integer.parseInt(updatedPrice),  
    updatedFlightInfo  
);  
String query = flight.getUpdateQuery(flight, Integer.parseInt(updatedFlightId));  
System.out.println(query);
```

```
@Override  
public String getUpdateQuery(Flight object, int id) {  
    return "UPDATE flight SET " +  
        "flight_id = " + object.getFlight_id() + ", " +  
        "guide_id = " + object.getGuide_id() + ", " +  
        "company_id = " + object.getCompany_id() + ", " +  
        "price = " + object.getPrice() + ", " +  
        "flight_info = '" + object.getFlight_info() +  
        "' WHERE flight_id = " + id;  
}
```

DELETE EXAMPLE - 1

Company deletes a hotel room

Room Entries

room_id	hotel_id	daily_price	room_properties
1	1	200	big
2	1	250	little

Insert Update Delete OK

Room Entries

room_id	hotel_id	daily_price	room_properties
1	1	200	big
2	1	250	little

Confirm Delete

Are you sure you want to delete this room?

Yes No

Insert Update Delete OK

Room Entries

room_id	hotel_id	daily_price	room_properties
1	1	200	big

Insert Update Delete OK

```
// Execute the necessary SQL DELETE command with the specified  
Room room = new Room();  
String query = room.getDeleteQuery(Integer.parseInt(roomId));  
System.out.println(query);
```

```
@Override  
public String getDeleteQuery(int id) {  
    return "DELETE FROM room WHERE room_id = " + id;  
}
```

DELETE EXAMPLE - 2

Company deletes a car
Enter company_id

View cars

Enter company_id:

1

OK Cancel

Before delete

```
Car car = new Car();  
String query = car.getDeleteQuery(Integer.parseInt(car_id.getText()));
```

Car Entries

car_id	company_id	car_properties	daily_price
1	1	Blue	200
2	1	Red	200
3	1	White	300

Car Entries

car_id	company_id	car_properties	daily_price
1	1	Blue	200
2	1	Red	200
3	1	White	300

Enter car info

car_id 2

OK Cancel

```
@Override  
public String getDeleteQuery(int id) {  
    return "DELETE FROM car WHERE car_id = " + id;  
}
```

Message

Delete successful

OK

After delete

Car Entries

car_id	company_id	car_properties	daily_price
1	1	Blue	200
3	1	White	300

DELETE EXAMPLE - 3

Deleting a guide
Enter company_id

View Guides

Enter company ID:

1

OK Cancel

View Guides

guide_id	company_id	person_id
1	1	2
2	1	5

View Guides

guide_id	company_id	person_id
1	1	2
2	1	5

Delete Guide

Are you sure you want to delete this guide?

Yes No

```
// Perform the necessary SQL delete operation  
String deleteQuery = new Guide().getDeleteQuery(Integer.parseInt(guideId));
```

Delete Guide

Guide deleted successfully.

OK

```
@Override  
public String getDeleteQuery(int id) {  
    return "DELETE FROM guide WHERE guide_id = " + id;  
}
```

After delete

View Guides

guide_id	company_id	person_id
1	1	2

FUNCTIONAL DEPENDENCIES

Table: accommodation

acc_id -> company_id, hotel_id, room_id, person_id, check_in_date, check_out_date

Table: bus

bus_id -> guide_id, company_id, price, trip_info

Table: car

car_id -> company_id, car_properties, daily_price

Table: car_rental

rental_id -> car_id, person_id, start_date, end_date

Table: company

company_id -> company_name

Table: flight

flight_id -> guide_id, company_id, price, flight_info

Table: guide

guide_id -> company_id, person_id

Table: hotel

hotel_id -> hotel_name, company_id

Table: person

person_id -> name, birthdate, budget

Table: room

room_id -> hotel_id, daily_price, room_properties

Table: transport

transport_id -> company_id, respective_id, person_id, transport_type

CHECKING NORMALIZATION

Table: accommodation

Functional dependencies: acc_id -> company_id, hotel_id, room_id, person_id, check_in_date, check_out_date

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and acc_id is a primary key.

Table: bus

Functional dependencies: bus_id -> guide_id, company_id, price, trip_info

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and bus_id is a primary key.

Table: car

Functional dependencies: car_id -> company_id, car_properties, daily_price

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and car_id is a primary key.

Table: car_rental

Functional dependencies: rental_id -> car_id, person_id, start_date, end_date

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and rental_id is a primary key.

Table: company

Functional dependencies: company_id -> company_name

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and company_id is a primary key.

Table: flight

Functional dependencies: flight_id -> guide_id, company_id, price, flight_info

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and flight_id is a primary key.

Table: guide

Functional dependencies: guide_id -> company_id, person_id

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and guide_id is a primary key.

Table: hotel

Functional dependencies: hotel_id -> hotel_name, company_id

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and hotel_id is a primary key.

Table: person

Functional dependencies: person_id -> name, birthdate, budget

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and person_id is a primary key.

Table: room

Functional dependencies: room_id -> hotel_id, daily_price, room_properties

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and room_id is a primary key.

Table: transport

Functional dependencies: transport_id -> company_id, respective_id, person_id, transport_type

This table does not violate BCNF or 3NF. The functional dependency does not have any trivial dependencies, and transport_id is a primary key.

Based on the evaluation, all the tables satisfy BCNF and 3NF. Therefore, no decomposition is required to achieve these normalization forms.

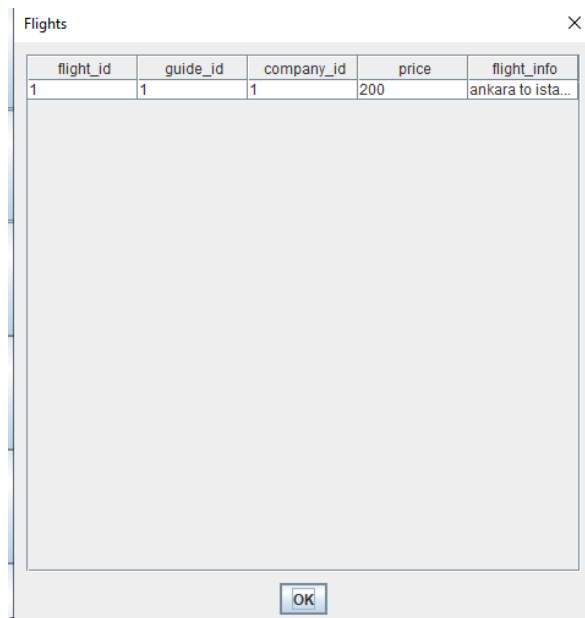
VIEWS

I've created my views like that :

```
try {
    st = TravelBookingConnection.getConnection().createStatement();
    st.execute( sql: "CREATE VIEW select_car AS" +
        " SELECT car_id,company_id,car_properties,daily_price FROM car;");
    st.execute( sql: "CREATE VIEW select_car_rental AS SELECT * FROM car_rental;");
    st.execute( sql: "CREATE VIEW select_accommodation AS SELECT * FROM accommodation;");
    st.execute( sql: "CREATE VIEW select_bus AS SELECT * FROM bus;");
    st.execute( sql: "CREATE VIEW select_flight AS SELECT * FROM flight;");
}
catch (SQLException ex) {
    ex.printStackTrace();
}
```

Interface example for a one view:

Person looks for flights:



The screenshot shows a Java Swing window titled "Flights" with a close button (X) in the top right corner. Inside the window is a table with the following data:

flight_id	guide_id	company_id	price	flight_info
1	1	1	200	ankara to ista...

Below the table is a large empty rectangular area, and at the bottom center is an "OK" button.

And queries for that views :

```
@Override
public String getSelectAllQuery() {
    return "SELECT * FROM select_car";
}
```

```
@Override
public String getSelectAllQuery() {
    return "SELECT * FROM select_car_rental";
}
```

```
@Override
public String getSelectAllQuery() { return "SELECT * FROM select_accommodation"; }
```

```
@Override
public String getSelectAllQuery() { return "SELECT * FROM select_bus"; }
```

```
@Override
public String getSelectAllQuery() { return "SELECT * FROM select_flight"; }
```

TRIGGERS

Update person budget when accommodation is bought or canceled

```
• CREATE DEFINER='root'@'localhost' TRIGGER `accommodation_AFTER_INSERT` AFTER INSERT ON `accommodation` FOR EACH ROW BEGIN
-- Retrieve the daily_price for the room from the room table
DECLARE room_price INT;
SELECT daily_price INTO room_price
FROM room
WHERE room_id = NEW.room_id AND hotel_id = NEW.hotel_id;

-- Update the person's budget
UPDATE person
SET budget = budget - (room_price * DATEDIFF(NEW.check_out_date, NEW.check_in_date))
WHERE person_id = NEW.person_id;
END
```

```
CREATE DEFINER='root'@'localhost' TRIGGER `accommodation_AFTER_DELETE` AFTER DELETE ON `accommodation` FOR EACH ROW BEGIN
-- Retrieve the daily_price for the room from the room table
DECLARE room_price INT;
SELECT daily_price INTO room_price
FROM room
WHERE room_id = OLD.room_id AND hotel_id = OLD.hotel_id;

-- Update the person's budget
UPDATE person
SET budget = budget + (room_price * DATEDIFF(OLD.check_out_date, OLD.check_in_date))
WHERE person_id = OLD.person_id;
END
```


Update person budget when car rental is bought or canceled

```
CREATE DEFINER='root'@'localhost' TRIGGER `car_rental_AFTER_INSERT` AFTER INSERT ON `car_rental` FOR EACH ROW BEGIN
    -- Retrieve the daily_price for the car from the Car table
    DECLARE car_price INT;
    SELECT daily_price INTO car_price FROM Car WHERE car_id = NEW.car_id;

    -- Update the person's budget
    UPDATE person
    SET budget = budget - (car_price * DATEDIFF(NEW.end_date, NEW.start_date))
    WHERE person_id = NEW.person_id;
END
```

```
CREATE DEFINER='root'@'localhost' TRIGGER `car_rental_AFTER_DELETE` AFTER DELETE ON `car_rental` FOR EACH ROW BEGIN
    -- Retrieve the car's daily_price from the Car table
    DECLARE car_price INT;
    SELECT daily_price INTO car_price FROM Car WHERE car_id = OLD.car_id;

    -- Update the person's budget
    UPDATE person
    SET budget = budget + (car_price * DATEDIFF(OLD.end_date, OLD.start_date))
    WHERE person_id = OLD.person_id;
END
```

Update person budget when a transport is bought or canceled

```
CREATE DEFINER='root'@'localhost' TRIGGER `transport_AFTER_INSERT` AFTER INSERT ON `transport` FOR EACH ROW BEGIN
    DECLARE transport_price INT;
```

```
    -- Retrieve the price based on the transport_type and respective_id
    IF NEW.transport_type = 'flight' THEN
        SELECT price INTO transport_price
        FROM flight
        WHERE flight_id = NEW.respective_id AND company_id = NEW.company_id;
    ELSEIF NEW.transport_type = 'bus' THEN
        SELECT price INTO transport_price
        FROM bus
        WHERE bus_id = NEW.respective_id AND company_id = NEW.company_id;
    END IF;
```

```
    -- Update the person's budget
    UPDATE person
    SET budget = budget - transport_price
    WHERE person_id = NEW.person_id;
END
```

```
CREATE DEFINER='root'@'localhost' TRIGGER `transport_AFTER_DELETE` AFTER DELETE ON `transport` FOR EACH ROW BEGIN
    DECLARE transport_price INT;
```

```
    -- Retrieve the price based on the transport_type and respective_id
    IF OLD.transport_type = 'flight' THEN
        SELECT price INTO transport_price
        FROM flight
        WHERE flight_id = OLD.respective_id AND company_id = OLD.company_id;
    ELSEIF OLD.transport_type = 'bus' THEN
        SELECT price INTO transport_price
        FROM bus
        WHERE bus_id = OLD.respective_id AND company_id = OLD.company_id;
    END IF;
```


```
    -- Update the person's budget
    UPDATE person
    SET budget = budget + transport_price
    WHERE person_id = OLD.person_id;
END
```

Person rents a
car :
Look for
person budget

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8400
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

Available Rentings



car_id	company_id	car_properties	daily_price
1	1	Blue	200
2	1	Red	200
3	1	Black	250

Car

Person ID:

Start date:

End date:

Budget is updated on trigger

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	6900
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

lets cancel the car rental so that budget is updated again with a trigger

Input

?

Enter Person ID:

1

OK

Cancel

Car Rental Entries for Person ID: 1

?

rental_id	car_id	person_id	start_date	end_date
10	3	1	2023-06-11	2023-06-17

OK

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8400
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

Insert

Update

Delete

OK

Can't rent a car more than one week and check budget is enough

```
CREATE DEFINER='root'@'localhost' TRIGGER `car_rental_BEFORE_INSERT` BEFORE INSERT ON `car_rental` FOR EACH ROW BEGIN
    DECLARE num_days INT;
    DECLARE total_cost INT;
    DECLARE person_budget INT;

    SET num_days = DATEDIFF(NEW.end_date, NEW.start_date);

    SELECT daily_price * num_days INTO total_cost
    FROM car
    WHERE car_id = NEW.car_id;

    SELECT budget INTO person_budget
    FROM person
    WHERE person_id = NEW.person_id;

    IF num_days > 7 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The duration of rental exceeds 7 days.';
    END IF;

    IF total_cost > person_budget THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The person does not have enough budget for rental.';
    END IF;
END
```

Try to rent a car

Car ×

Person ID:

2

Start date:

2023-06-23

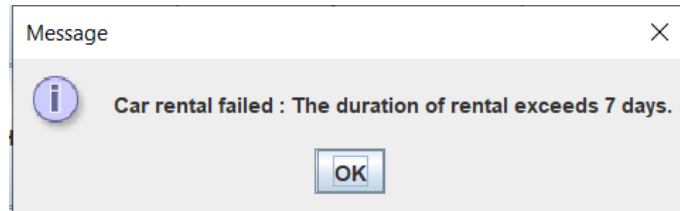
End date:

2023-07-23

OK

Cancel

Message ×



Car rental failed : The duration of rental exceeds 7 days.

OK

Car ×

Person ID:

2

Start date:

2023-06-06


End date:

2023-06-08

OK

Cancel

Message ×



Car rental failed : The person does not have enough budget for rental.

OK

Cant accommodate on a hotel more than one month and check budget is enough

```
CREATE DEFINER='root'@'localhost' TRIGGER `accommodation_BEFORE_INSERT` BEFORE INSERT ON `accommodation`  
FOR EACH ROW BEGIN  
    DECLARE num_days INT;  
    DECLARE total_cost INT;  
    DECLARE person_budget INT;  
  
    SET num_days = DATEDIFF(NEW.check_out_date, NEW.check_in_date);  
  
    SELECT daily_price * num_days INTO total_cost  
    FROM room  
    WHERE room_id = NEW.room_id AND hotel_id = NEW.hotel_id;  
  
    SELECT budget INTO person_budget  
    FROM person  
    WHERE person_id = NEW.person_id;  
  
    IF num_days > 30 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'The duration of stay exceeds 30 days.';  
    END IF;  
  
    IF total_cost > person_budget THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'The person does not have enough budget for accommodation.';  
    END IF;  
END
```

Accommodate


Person ID:

Check in date:

Check out date:

OK Cancel

Message

 Accommodation failed : The person does not have enough budget for accommodation.

OK

Accommodate


Person ID:

Check in date:

Check out date:

OK Cancel

Message

 Accommodation failed : The duration of stay exceeds 30 days.

OK

Check budget for transportation

```
CREATE DEFINER='root'@'localhost' TRIGGER `transport_BEFORE_INSERT` BEFORE INSERT ON `transport` FOR EACH ROW BEGIN
    DECLARE total_cost INT;
    DECLARE person_budget INT;
    DECLARE respective_id INT;
    DECLARE tr_type VARCHAR(255);

    SELECT budget INTO person_budget
    FROM person
    WHERE person_id = NEW.person_id;

    SET tr_type = NEW.transport_type;
    SET respective_id = NEW.respective_id;

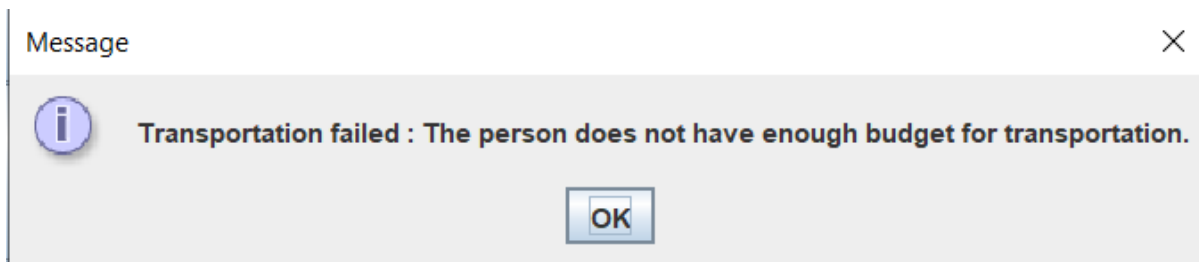
    IF tr_type = "flight" THEN
        SELECT price INTO total_cost
        FROM flight
        WHERE respective_id = flight_id;

    END IF;

    IF tr_type = "bus" THEN
        SELECT price INTO total_cost
        FROM bus
        WHERE respective_id = bus_id;

    END IF;

    IF total_cost > person_budget THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The person does not have enough budget for transportation.';
    END IF;
END
```



JOINS

RIGHT JOIN

While company is viewing car rentals :

```
String query = "SELECT DISTINCT car_rental.rental_id, car_rental.car_id, car_rental.person_id, car_rental.start_date, car_rental.end_date " +  
"FROM car_rental " +  
"RIGHT JOIN car ON car.car_id = car_rental.car_id " +  
"WHERE car.company_id = " + company_id;
```

Car Rental Entries

×

rental_id	car_id	person_id	start_date	end_date
11	3	1	2023-06-11	2023-06-17

Insert

Update

Delete

OK

LEFT JOIN

While company is viewing rooms of a hotel

```
String query = "SELECT room.room_id, room.hotel_id, room.daily_price, room.room_properties " +  
    "FROM room " +  
    "LEFT JOIN hotel ON room.hotel_id = hotel.hotel_id " +  
    "WHERE room.hotel_id = "+hotelID;
```

Room Entries ✕

room_id	hotel_id	daily_price	room_properties
1	1	200	big

Insert

Update

Delete

OK

FULL JOIN

My MySQL version does not support FULL JOIN keyword, I used this instead :
Person is viewing all hotels and all rooms

```
String query = "SELECT *\n" +  
    "FROM company\n" +  
    "LEFT JOIN flight ON company.company_id = flight.company_id\n" +  
    "LEFT JOIN car ON company.company_id = car.company_id\n" +  
    "LEFT JOIN bus ON company.company_id = bus.company_id\n" +  
    "UNION\n" +  
    "SELECT *\n" +  
    "FROM company\n" +  
    "RIGHT JOIN flight ON company.company_id = flight.company_id\n" +  
    "RIGHT JOIN car ON company.company_id = car.company_id\n" +  
    "RIGHT JOIN bus ON company.company_id = bus.company_id\n" +  
    "WHERE company.company_id =" + companyID ;
```

Hotels And Rooms

co...	co...	flig...	gui...	co...	pri...	flig...	car...	co...	car...	dai...	bu...	gui...	co...	price	trip...
1	C...	1	1	1	200	an...	3	1	W...	300	1	1	1	200	ist...
1	C...	1	1	1	200	an...	2	1	Red	200	1	1	1	200	ist...
1	C...	1	1	1	200	an...	1	1	Blue	200	1	1	1	200	ist...

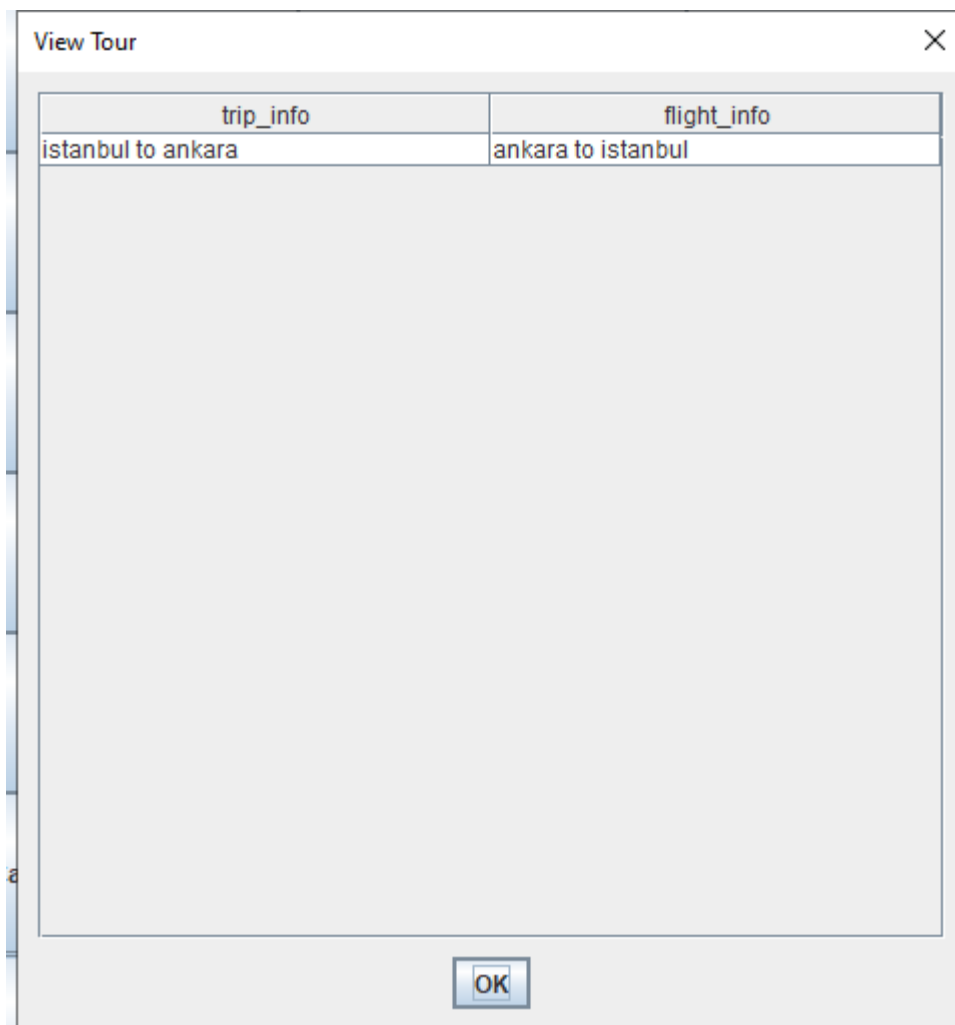
OK

see
and

INNER JOIN

When person wants to
the companies flight
bus infos together

```
try {  
    // Prompt the user to enter the company ID  
    String companyID = JOptionPane.showInputDialog( parentComponent: this, m  
        JOptionPane.QUESTION_MESSAGE);  
  
    String query = "SELECT bus.trip_info,flight.flight_info\n" +  
        "FROM bus\n" +  
        "INNER JOIN flight ON bus.company_id = flight.company_id;";
```



ATOMIC TRANSACTIONS

While person adds a budget

DDL:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `para_transfer`(IN personid INT,IN amount INT)
2 BEGIN
3 START TRANSACTION ;
4 UPDATE person SET budget = budget + amount WHERE person_id = personid;
5 COMMIT;
6 END
```

Add Budget

Enter person ID:

1

OK Cancel

Enter amount

Your budget is : 8400

Enter amount to add : 100

OK Cancel

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8500
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

Insert Update Delete

OK

While person removes a budget

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `para_cek`(IN personid INT,IN amount INT)
BEGIN
START TRANSACTION ;
UPDATE person SET budget = budget - amount WHERE person_id = personid;
COMMIT;
END
```

Remove Budget

Enter person ID:

1

OK Cancel

Enter amount

Your budget is : 8500

Enter amount to remove : 100

OK Cancel

Person Entries

person_id	name	birthdate	budget
1	Talha	2000-07-12	8400
2	Yakup	2000-07-14	2000
3	Jacob	2000-07-16	2000
4	John	2000-07-18	2000

Insert Update Delete

OK

While company transfers a guide

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `transfer_guide`(IN guideID INT,IN newCompanyID INT)
2 BEGIN
3     -- Start the transaction
4     START TRANSACTION;
5
6     -- Update the guide's company_id to the newCompanyID
7     UPDATE guide
8     SET company_id = newCompanyID
9     WHERE guide_id = guideID;
10
11     -- Check if the update was successful
12     IF ROW_COUNT() = 0 THEN
13         -- Rollback the transaction if the guide ID was not found
14         ROLLBACK;
15         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Guide ID not found.';
16     ELSE
17         -- Commit the transaction
18         COMMIT;
19     END IF;
20 END
```

View Guides

Enter company ID:

1

OK Cancel

View and transfer the guide.

Transfer Guide

Guide ID: 2

New Company ID: 2

OK Cancel

Transfer Guide

Guide transferred successfully.

OK

Transfer Guides

guide_id	company_id	person_id
1	1	2
2	1	3

Transfer

OK

View Guides

Enter company ID:

2

OK Cancel

After transferring :

View Guides

guide_id	company_id	person_id
2	2	3

While company transfers a car_rental

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `transfer_car_rental`(IN oldPersonID INT, IN newPersonID INT)
2 BEGIN
3     DECLARE oldPersonBudget DECIMAL(10, 2);
4     DECLARE newPersonBudget DECIMAL(10, 2);
5     DECLARE rentalID INT;
6     DECLARE dailyPrice INT;
7     DECLARE startDate DATE;
8     DECLARE endDate DATE;
9     DECLARE carId INT;
10    DECLARE rentalCost INT;
11
12    -- Start the transaction
13    START TRANSACTION;
14
15    -- Check if the old person exists and get their budget
16    SELECT budget INTO oldPersonBudget FROM person WHERE person_id = oldPersonID;
17    IF oldPersonBudget IS NULL THEN
18        -- Rollback the transaction if the old person does not exist
19        ROLLBACK;
20        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Old person not found.';
21    END IF;
22
23    -- Check if the new person exists and get their budget
24    SELECT budget INTO newPersonBudget FROM person WHERE person_id = newPersonID;
25    IF newPersonBudget IS NULL THEN
26        -- Rollback the transaction if the new person does not exist
27        ROLLBACK;
28        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'New person not found.';
29    END IF;
30
31
32    SELECT rental_id,start_date,end_date,car_id INTO rentalID,startDate,endDate,carId FROM car_rental WHERE person_id = oldPersonID;
33    IF rentalID IS NULL THEN
34        -- Rollback the transaction if the car rental does not exist for the old person
35        ROLLBACK;
36        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Car rental not found for the old person.';
37    END IF;
38
39    SELECT daily_price INTO dailyPrice FROM car WHERE car_id = carId;
40    SET rentalCost = (dailyPrice * DATEDIFF(startDate, endDate));
41    IF newPersonBudget < rentalCost THEN
42        -- Rollback the transaction if the new person's budget is insufficient
43        ROLLBACK;
44        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Insufficient budget for the transfer.';
45    END IF;
46
47    -- Update the old person's budget by deducting the rental cost
48    UPDATE person SET budget = budget - rentalCost WHERE person_id = oldPersonID;
49
50    -- Update the car_rental record with the new person's ID
51    UPDATE car_rental SET person_id = newPersonID WHERE rental_id = rentalID;
52
53    -- Check if the update was successful
54    IF ROW_COUNT() = 0 THEN
55        -- Rollback the transaction if the rental ID was not found
56        ROLLBACK;
57        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Rental ID not found.';
58    ELSE
59        -- Commit the transaction
60        COMMIT;
61    END IF;
```

Tranfer Car Rental

?

Enter personID:

2

OK

Cancel

Transfer Car Rental

rental_id	car_id	person_id	start_date	end_date
11	3	2	2023-06-11	2023-06-17

Transfer

OK

Transfer:

Transfer Car Rental

Old Person ID:

2

New Person ID:

1

OK

Cancel

Updated table :

Transfer Car Rental

i

Car Rental transferred successfully.

OK

Car Rental Entries

rental_id	car_id	person_id	start_date	end_date
11	3	1	2023-06-11	2023-06-17